

AppVet

Version 1.01

July 2014

Stephen Quirolgico
Tom Karygiannis
Jeff Voas



Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology

Contents

1	INTRODUCTION	1
1.1	Clients	2
1.2	Tools	2
1.2.1	Synchronous Tool Service.....	2
1.2.2	Asynchronous Tool Service.....	3
1.2.3	Push Tool Service.....	3
1.3	NIST Software Agreement.....	5
2	SYSTEM REQUIREMENTS	6
2.1	Hardware Requirements.....	6
2.2	Platform Requirements	6
2.2.1	Java	6
2.2.2	Android APKTool (Android only)	6
2.2.3	Apache Tomcat.....	6
2.2.4	MySQL	7
3	INSTALLATION.....	8
3.1	AppVet Installer.....	8
3.2	AppVet Configuration.....	11
3.3	Tool Service Configuration.....	12
4	USER'S GUIDE	13
4.1	Launching AppVet.....	13
4.2	App Management Interface.....	13
4.2.1	Operation Buttons.....	14
4.2.2	Menus	15
4.2.3	Tool Status.....	15
4.2.4	App Status.....	16
4.3	Managing Tools	17
5	DEVELOPER'S GUIDE.....	18
5.1	Eclipse.....	18
5.1.1	Integrating Tomcat.....	18
5.1.2	Downloading AppVet.....	18
5.1.3	Importing AppVet.....	19
5.1.4	AppVet Compilation.....	19
5.1.5	Installing AppVet.....	19
5.1.6	Launching AppVet.....	19
5.1.7	Exporting AppVet.....	19
5.2	AppVet Architecture	19
5.2.1	AppVet Servlet	20
5.2.2	GWT Client/Server.....	20
5.2.3	Tool Manager	20

5.2.4	Other	21
5.3	AppVet Source Code Files.....	21
5.4	Database.....	21
5.5	Tool Services	22
APPENDIX A: APPVET API		23
A.1	AUTHENTICATE	23
A.1.1	HTTP Request	23
A.1.2	HTTP Response.....	23
A.2	GET_STATUS	23
A.2.1	HTTP Request	23
A.2.2	HTTP Response.....	23
A.3	GET_TOOL_REPORT	24
A.3.1	HTTP Request	24
A.3.2	HTTP Response.....	24
A.4	GET_APP_LOG.....	24
A.4.1	HTTP Request	24
A.4.2	HTTP Response.....	24
A.5	GET_APPVET_LOG	25
A.5.1	HTTP Request	25
A.5.2	HTTP Response.....	25
A.6	DOWNLOAD_REPORTS	25
A.6.1	HTTP Request	25
A.6.2	HTTP Response.....	25
A.7	SUBMIT_APP.....	26
A.6.1	HTTP Request	26
A.6.2	HTTP Response.....	26
A.8	SUBMIT_REPORT	26
A.6.1	HTTP Request	26
A.6.2	HTTP Response.....	26
APPENDIX B: TOOL SERVICE API REQUIREMENTS		27
B.1	General Requirements.....	27
B.1.1	HTTP Request.....	27
B.1.2	Additional Information	27
B.2	Synchronous Response	27
B.3	Asynchronous Request	28
B.4	Asynchronous and Push Reports	28
APPENDIX C: APPVET PROPERTIES SCHEMA.....		29
APPENDIX D: TOOL SERVICE ADAPTER SCHEMA		33
APPENDIX E: APPVET DATABASE.....		38
E.1	Table users	38
E.2	Table sessions	38
E.3	Table apps.....	39
E.4	Table status	40
APPENDIX F: CHANGE HISTORY		41
7/14/14.....		41
5/28/14.....		41

List of Figures

Figure 1-1. AppVet.....	1
Figure 1-2. AppVet system architecture.....	2
Figure 1-3. Synchronous tool service protocol.....	3
Figure 1-4. Asynchronous tool service protocol.	4
Figure 1-5. Push tool service protocol.....	4
Figure 3-1. AppVet installer.....	8
Figure 3-2. Host dialog.....	9
Figure 3-3. Admin dialog.	10
Figure 3-4. Platform dialog.	10
Figure 3-5. AppVet installation progress.	11
Figure 3-6. AppVet files directory.	11
Figure 4-1. AppVet login screen.	13
Figure 4-2. AppVet app management interface.....	14
Figure 4-3. User settings.....	15
Figure 4-4. Tool statuses.	15
Figure 4-5. App status.	16
Figure 5-1. AppVet Architecture.....	20
Figure 5-2. AppVet source code release.....	21

List of Tables

Table 4-1. AppVet operation buttons.	14
Table 4-2. Tool status and risk assessment descriptions.	16
Table 4-3. App status descriptions.	17

Code Listings

Listing C-1. AppVetProperties XML Schema.	29
Listing D-1. Tool Service Adapter Schema.....	33

1 INTRODUCTION

AppVet is a simple web-based application for vetting mobile apps. It facilitates the app vetting workflow by providing an intuitive user interface for submitting and testing apps, accessing reports, and assessing risk. AppVet is designed to easily and seamlessly integrate with a wide variety of third-party tools including static and dynamic analyzers, anti-virus scanners, and vulnerability repositories through the specification of simple APIs and requirements. AppVet also supports easy and seamless integration with clients including app stores and continuous integration environments. AppVet is shown in Figure 1-1.

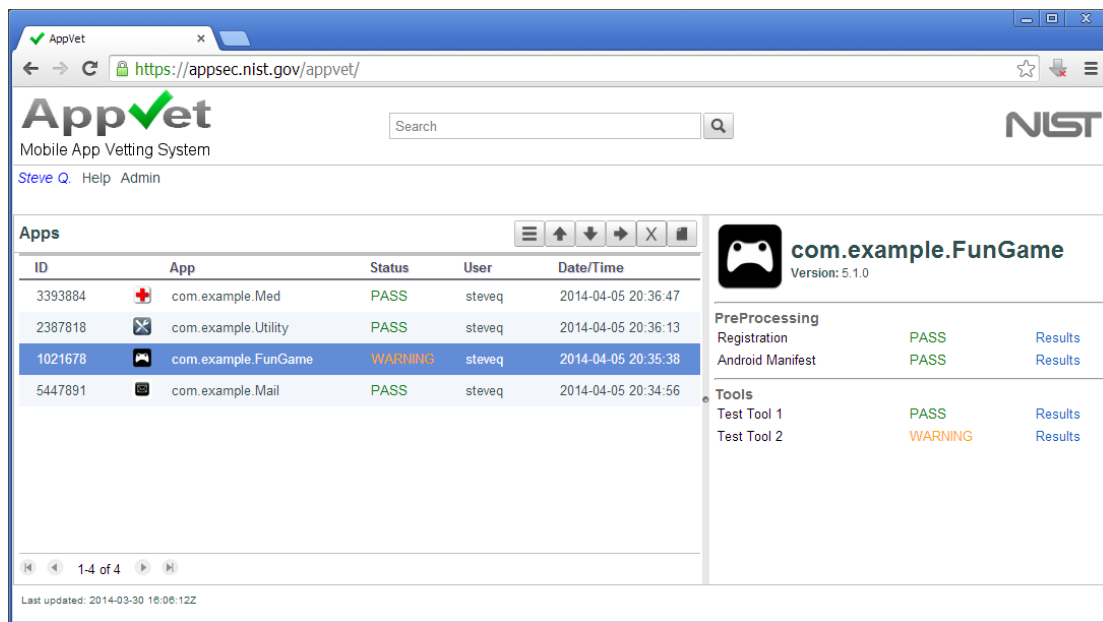


Figure 1-1. AppVet.

An AppVet system comprises an AppVet web application and its related tools and clients. In an AppVet system, the app vetting workflow begins when a client submits an app to AppVet. When AppVet receives an app, it registers the app and performs some pre-processing of the app. Preprocessing is used to extract meta-data about an app and possibly provide additional functionality such as ensuring that the app conforms to specific requirements of the hosting organization. After preprocessing an app, AppVet sends the app and related information to one or more tools for testing and evaluation. When a tool completes its analysis, it returns a report and risk assessment to AppVet which, in turn, makes them available to clients. In addition, AppVet generates an overall risk assessment based on risk assessments from all tools. The AppVet system architecture is shown in Figure 1-2.

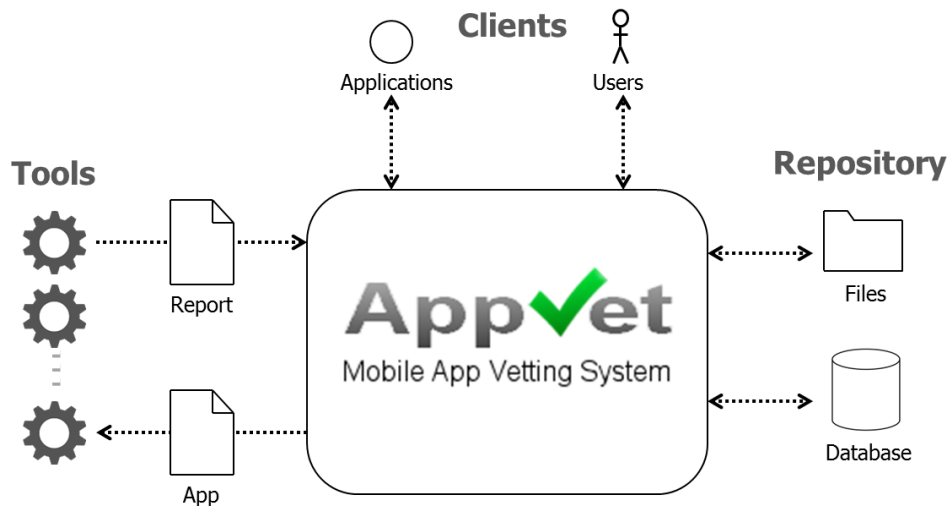


Figure 1-2. AppVet system architecture.

1.1 Clients

AppVet relies on clients to submit apps to the system and to consume reports and risk assessments. Clients include users (such as developers and analysts) and applications (such as app stores and continuous integration environments). Using a web browser, users can access AppVet via the AppVet app management interface. This interface provides support for uploading apps, accessing reports and assessing risk. Applications access AppVet through the AppVet API as described in Appendix A.

1.2 Tools

AppVet relies on tools to test and evaluate apps.¹ A tool is provided by a third-party vendor, tool developer, or user that wants to leverage an existing tool. In an AppVet system, tools are made available as online services called *tool services*. To facilitate integration of tool services with AppVet, AppVet requires these services to implement a simple REST API for submitting apps to the service and for acquiring reports and risk assessments from the service. This API must also conform to additional requirements as described in Appendix B, Tool Service API Requirements. AppVet defines three types of tool services: *synchronous*, *asynchronous*, and *push* tool services.

1.2.1 Synchronous Tool Service

A synchronous tool service is a service that responds to an AppVet HTTP Request message with a report and risk assessment via the corresponding HTTP Response message. Synchronous tool services are aimed at analyses that can be performed relatively quickly on an app. Figure 1-3 shows the AppVet synchronous tool service protocol.

¹Note that neither the AppVet binary nor source code release include tools. Tools must be added and configured separately by the AppVet administrator.

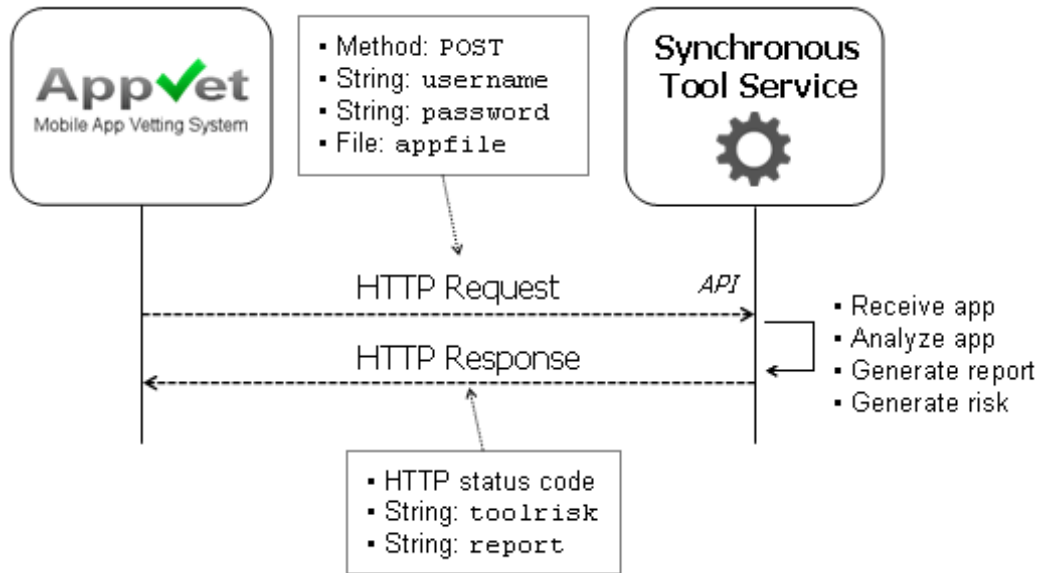


Figure 1-3. Synchronous tool service protocol.

The AppVet API requirements for synchronous tool services are described in Appendix B. For tool services that take an extended amount of time to test and evaluate an app, an asynchronous tool service type may be used.

1.2.2 Asynchronous Tool Service

An asynchronous tool service is a service that responds to an AppVet HTTP Request with a report and risk assessment via a new HTTP Request to AppVet at some unspecified point in the future. Figure 1-4 shows the AppVet asynchronous tool service protocol. For more details about asynchronous services see Appendix B.3, Asynchronous Request-Response, and Appendix B.4, Asynchronous and Push Reports.

1.2.3 Push Tool Service

A push tool service is a service that sends a report and risk assessment to AppVet without first receiving a corresponding request from AppVet. Such cases occur, for example, if the service analyzes an app on behalf of another tool service. After the tool service analyzes the app, it sends the report and risk assessment to AppVet as shown in Figure 1-5. For more details about push services see Appendix B.4, Asynchronous and Push Reports.

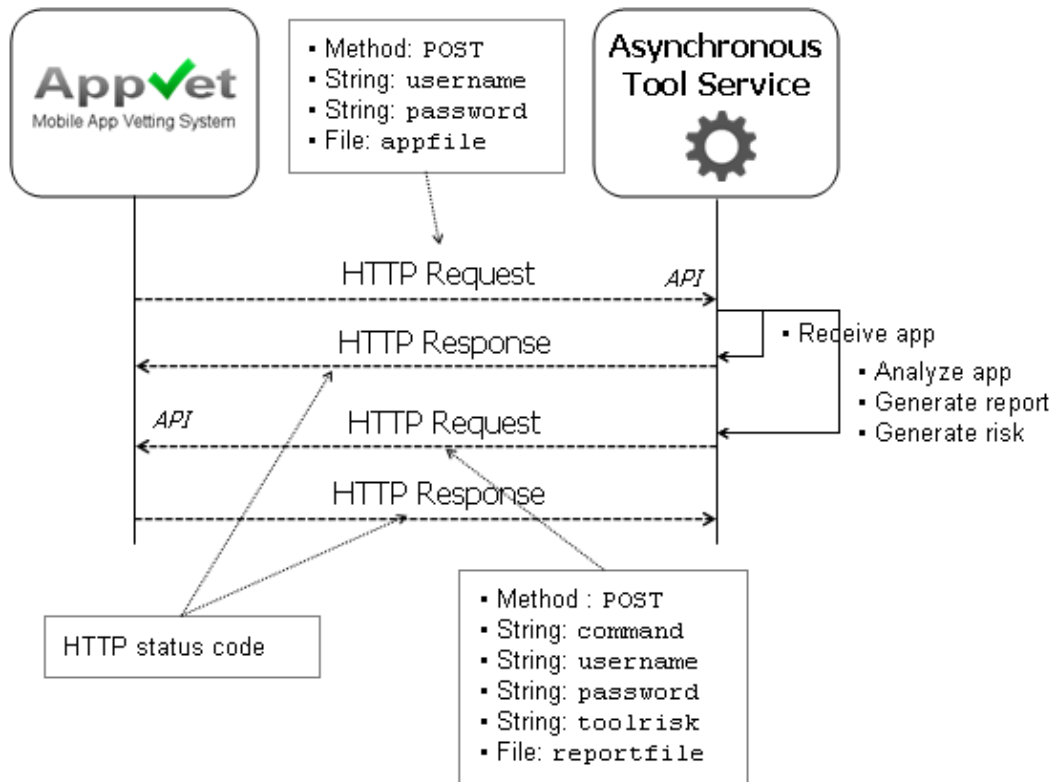


Figure 1-4. Asynchronous tool service protocol.

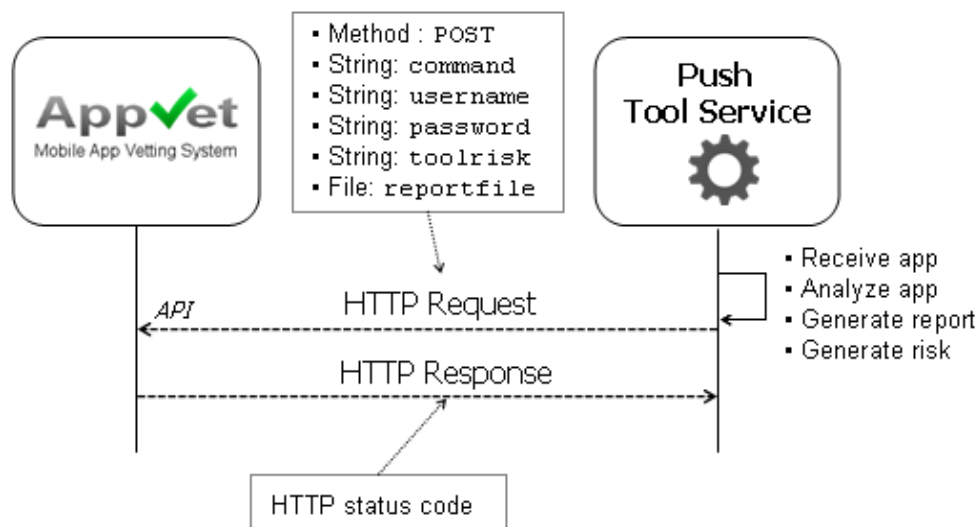


Figure 1-5. Push tool service protocol.

1.3 NIST Software Agreement

This software was developed by employees of the National Institute of Standards and Technology (NIST), an agency of the Federal Government. Pursuant to title 15 United States Code Section 105, works of NIST employees are not subject to copyright protection in the United States and are considered to be in the public domain. As a result, a formal license is not needed to use the software.

This software is provided by NIST as a service and is expressly provided "AS IS". NIST MAKES NO WARRANTY OF ANY KIND, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT AND DATA ACCURACY. NIST does not warrant or make any representations regarding the use of the software or the results thereof including, but not limited to, the correctness, accuracy, reliability or usefulness of the software.

Permission to use this software is contingent upon your acceptance of the terms of this agreement.

2 SYSTEM REQUIREMENTS

2.1 Hardware Requirements

AppVet has the following minimum hardware requirements:

- 64-bit Windows recommended (Note that although AppVet is Java-based, installation problems have been reported under Linux)
- 512MB RAM
- 1GB free hard disk space
- Network access with static IP address

2.2 Platform Requirements

AppVet requires the following to be installed on the host.

2.2.1 Java

AppVet requires Java JDK 7 to be installed. For 64-bit operating systems, it is recommended to use 64-bit Java JDK. After installing Java JDK, set the `JAVA_HOME` environment variable to the path of your Java JDK root directory.

2.2.2 Android APKTool (Android only)

For Android app analysis, AppVet requires Android APKTool 2.0 to be installed. After installing APKTool, add the path of your APKTool root directory to your `PATH` environment variable. Ensure that APKTool can be executed from a console (e.g., `apktool.bat` on Windows).

2.2.3 Apache Tomcat

AppVet requires a web server and servlet container to be installed. Although AppVet can use any web server or servlet container, this document describes the use of Apache Tomcat 7. For 64-bit operating systems, it is recommended to use 64-bit Tomcat. For Windows systems, it is recommended to use the Apache Tomcat 32-bit/64-bit Windows Service Installer to install Tomcat. After installing Tomcat, set the `CATALINA_HOME` environment variable to the path of your Tomcat root directory. Note that using the Windows Service Installer will automatically set `CATALINA_HOME` during installation.

To manage Tomcat, add a user with attribute `roles="manager-gui"` to the `$CATALINA_HOME/conf/tomcat-users.xml` configuration file. This will allow the user to manage Tomcat from the Tomcat web browser UI.

Verify proper installation of Tomcat by starting the Tomcat service, opening a browser, and ensuring the Tomcat page is visible at `http://<host>:<port>`. Also ensure that Tomcat starts automatically after reboot.

AppVet recommends the use of Secure Socket Layer (SSL). To enable SSL for Apache Tomcat, add an SSL-enabled connector to the `$CATALINA_HOME/conf/server.xml` configuration file. For example:

```
<Connector SSLEnabled="true" clientAuth="false"
keystoreFile="C:\mykeystore.jks"
keystorePass="mypassword123!" maxThreads="150"
port="443" protocol="HTTP/1.1" scheme="https"
secure="true" sslProtocol="TLS"/>
```

Note that when using SSL, a keystore containing a valid certificate should reside on the host. Note that for production systems, a certificate from a Certificate Authority (CA) should be used in lieu of a self-signed certificate.

2.2.4 MySQL

AppVet requires a relational database to be installed. Although AppVet can use any relational database that supports JDBC, this document describes the use of MySQL 5 Community Server Edition. For 64-bit operating systems, it is recommended to use 64-bit MySQL.

3 INSTALLATION

AppVet installation involves:

- Creating MySQL database and tables.
- Adding an AppVet administrator account to the database.
- Creating the \$APPVET_FILES_HOME directory and related AppVet files.
- Copying the AppVet Web Application archive (WAR) file to Tomcat.

For specifications of the AppVet database and tables, see Appendix E, Database Schemas.

3.1 AppVet Installer

Set the environment variable APPVET_FILES_HOME to a directory path on the host system. It is recommended to use a path that does not contain spaces or special characters. For example, use C:\appvet_files on Windows.

Next, download the appvet_1.0_installer.zip file and extract it to your host system. Open a console and change directory to the extracted AppVet installer directory /appvet_1.0_installer. From the console, run the AppVet installer using:

```
> java -jar AppVetInstaller.jar
```

Here, you should see the AppVet installer application as shown in Figure 3-1.

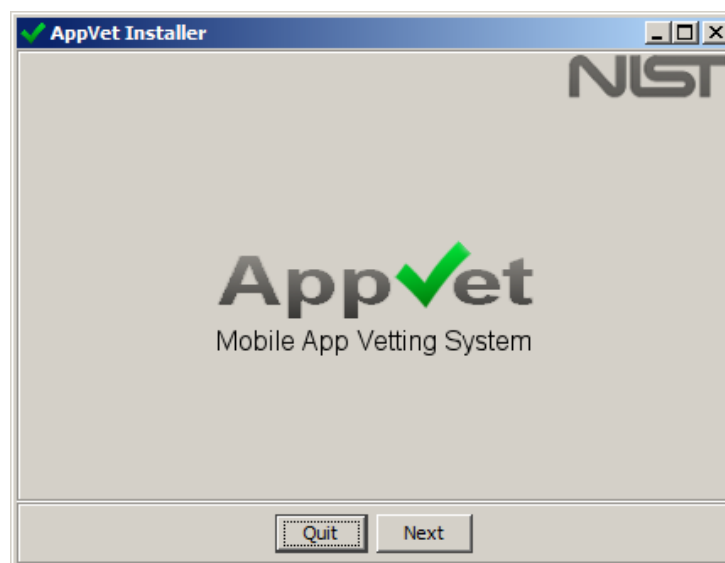


Figure 3-1. AppVet installer.

Select **Next** to view the Host dialog as shown in Figure 3-2.

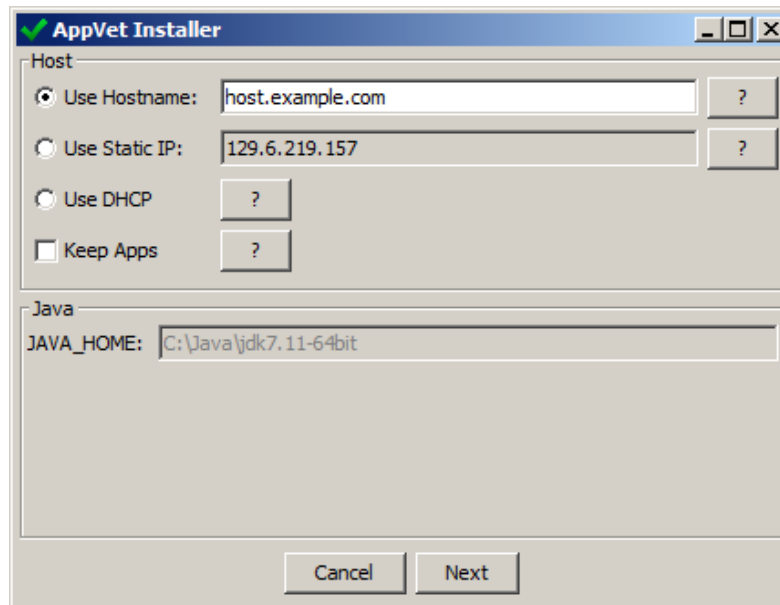


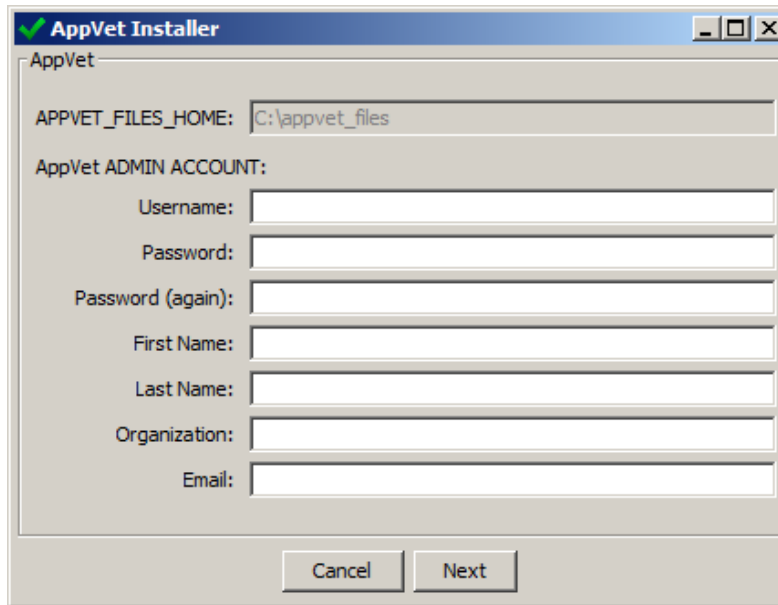
Figure 3-2. Host dialog.

The Host dialog provides the following options:

- **Use Hostname:** This option should be selected for production systems or development systems that have a fully-qualified domain name (FQDN).
- **Use Static IP:** This option should be selected for development systems if a static IP is available. This option should not be selected for production systems.
- **Use DHCP:** This option should be selected for development systems using a dynamic IP address.
- **Keep Apps:** This option should be selected if apps should be archived on the system. If this option is not selected, apps will be deleted after they are processed. Note that keeping apps on the system may result in significant disk space usage.

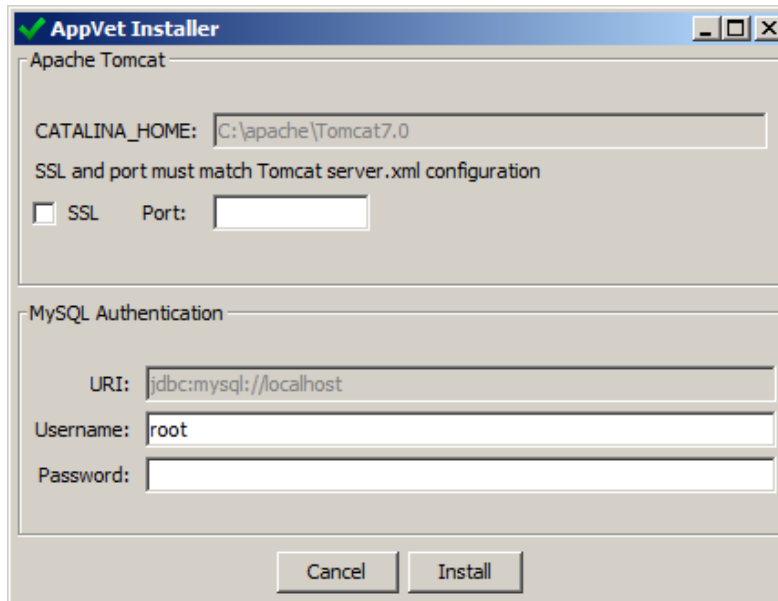
Select **Next** to view the Administrator dialog as shown in Figure 3-3. Enter the AppVet administrator account information. Note that additional administrators can be added later. Select **Next** to view the Platform dialog as shown in Figure 3-4.

Select **SSL** if Secure Socket Layer is enabled for your Tomcat server. In addition, center the primary port of your Tomcat server. Note that this information must match your Tomcat `server.xml` configuration. Next, add your MySQL username and password. Select **Install** to view the AppVet installation progress as shown in Figure 3-5.



The 'AppVet Installer' window displays the 'AppVet' configuration page. It includes a text field for 'APPVET_FILES_HOME' with the value 'C:\appvet_files'. Below this is the 'AppVet ADMIN ACCOUNT' section, which contains input fields for 'Username', 'Password', 'Password (again)', 'First Name', 'Last Name', 'Organization', and 'Email'. At the bottom of the window are 'Cancel' and 'Next' buttons.

Figure 3-3. Admin dialog.



The 'AppVet Installer' window displays the 'Apache Tomcat' configuration page. It features a 'CATALINA_HOME' text field with the value 'C:\apache\Tomcat7.0'. Below this is a note: 'SSL and port must match Tomcat server.xml configuration'. There is an unchecked checkbox for 'SSL' and a 'Port' text field. The 'MySQL Authentication' section contains fields for 'URI' (value: 'jdbc:mysql://localhost'), 'Username' (value: 'root'), and 'Password'. At the bottom are 'Cancel' and 'Install' buttons.

Figure 3-4. Platform dialog.

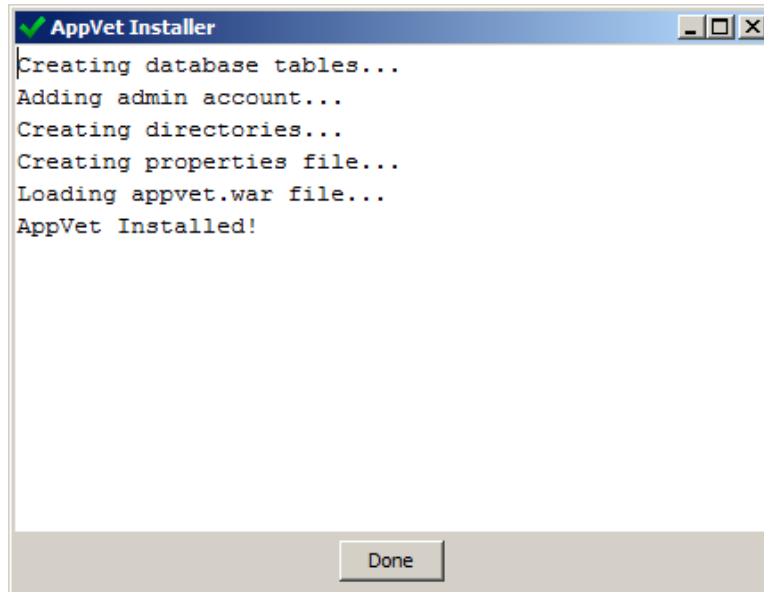


Figure 3-5. AppVet installation progress.

Files created in `$APPVET_FILES_HOME` are shown in Figure 3-6.

```

$APPVET_FILES_HOME
├── /apps - Contains received apps and related reports (initially empty)
├── /conf
│   ├── AppVetProperties.xml - AppVet configuration file
│   └── /tool_adapters
│       ├── appinfo.xml - Android meta-data configuration file
│       ├── registration.xml - AppVet registration configuration file
│       └── android-cert-test.xml - Example tool service adapter
├── /logs
│   └── appvet_log.txt - AppVet log

```

Figure 3-6. AppVet files directory.

3.2 AppVet Configuration

When AppVet is installed, it is configured with default values that should be appropriate in most cases. The AppVet configuration file `AppVetProperties.xml` is located in the `$APPVET_FILES_HOME/conf` directory. The specification for `AppVetProperties.xml` is defined in Appendix C, AppVet Properties Schema.

3.3 Tool Service Configuration

AppVet accesses a tool service using a *tool service adapter*. A tool service adapter is an XML configuration file that defines properties of a tool service including the service's required (REST) API parameters. A tool service adapter's structure is defined in Appendix D, Tool Service Adapter Schema. Typically, tool service adapters will be provided by the tool vendor, developer, or user that provides a tool service for an existing tool. Thus, tool service adapters require little or no configuration by the AppVet administrator.

Adding a new tool service to AppVet involves adding a new tool service adapter to the `$APPVET_HOME/conf/tool_adapters` directory. When adding a new tool configuration file, AppVet automatically adds a new entry for the tool into the database. Note that newly added tools are not applied to previously processed apps. Instead, a tool status of N/A is displayed for such apps.

Tool services can be removed from AppVet by removing their corresponding adapter from the `$APPVET_HOME/conf/tool_adapters` directory. Removing a tool from AppVet will hide results for the tool even if an app has been previously processed by the tool. However, AppVet will continue to store all previously generated results for the tool in its repository. If the tool is later re-added to the system, AppVet will display the tool's results for previously processed apps. To ensure proper operation of AppVet, ensure that Tomcat is shut down before adding or removing a tool service adapter.

The AppVet administrator must ensure that tool service adapters for all desired tools are present in `$APPVET_HOME/conf/tool_adapters`. *Do not include adapters for tool services that are not available to AppVet or that AppVet cannot authenticate to. Doing so will lead to AppVet system errors for those tools.*

4 USER'S GUIDE

4.1 Launching AppVet

To launch AppVet, start the Tomcat server and open a browser to the AppVet URL:
`https://<host>:<port>/appvet` where `<host>:<port>` is the hostname or IP address and port number of your server. The AppVet login screen should be visible as shown in Figure 4-1.

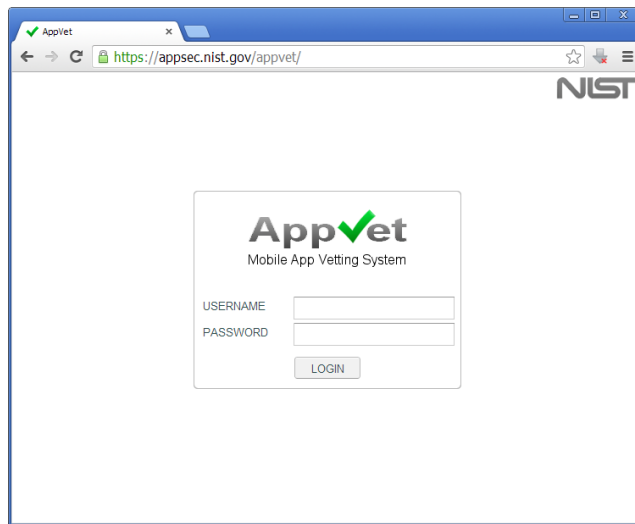


Figure 4-1. AppVet login screen.

4.2 App Management Interface

After logging into AppVet, the AppVet app management interface is displayed as shown in Figure 4-2. The AppVet app management interface comprises two main panels: an *apps list panel* on the left and an *app information panel* on the right. The apps list panel displays apps that have been uploaded to the system while the app information panel displays information about the selected app. The apps list panel displays general information about uploaded apps including their AppVet-generated app ID, app name, current status and risk assessment, user (app owner), and the date/time when the app was uploaded to the system. The app info panel contains information about a selected app including:

- App name and icon
- Version number
- Registration and app pre-processing statuses
- Tool service reports
- Tool service status and risk assessments

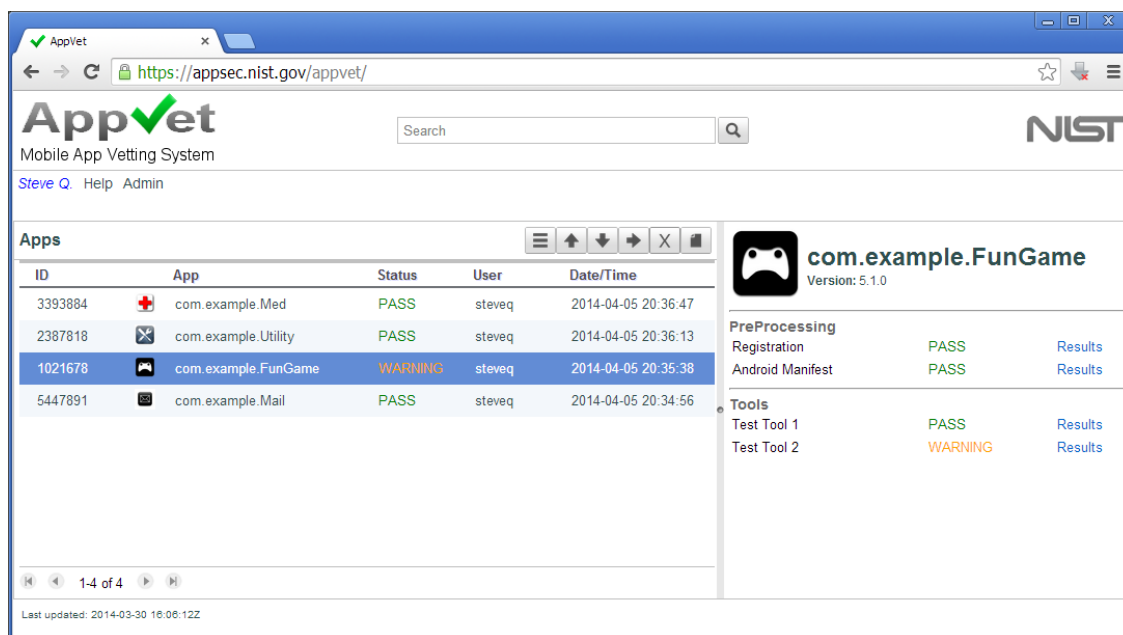


Figure 4-2. AppVet app management interface.

4.2.1 Operation Buttons

The apps list panel contains operation buttons used to manage apps and their related reports. Table 4-1 describes the functions of AppVet operation buttons.

Table 4-1. AppVet operation buttons.

Icon	Name	Description
	View All	View all apps.
	Upload	Upload app file.
	Download	Download app, report, or TA Release Kit reports for selected app.
	Set Report	Set (or override) report and risk assessment.
	Delete App	Delete selected app.
	View Log	View log for selected app.

4.2.2 Menus

AppVet provides three menus: User settings, Help, and Admin. The User settings menu is displayed as the user's first name and first letter of the last name in the top left corner as shown in Figure 4-3.

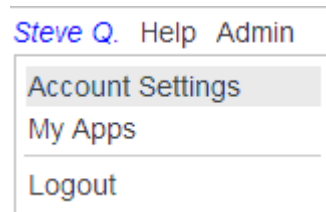


Figure 4-3. User settings.

The User settings menu allows users to reset their account setting information (via Account Settings), view apps that they have uploaded (via My Apps), and log out (via Logout).

The Help menu provides help-related information.

For AppVet administrators, the Admin menu allows viewing of the AppVet system log as well as ability to manage AppVet users.

4.2.3 Tool Status

In the app information panel, each tool service is associated with a processing status or risk assessment as shown in Figure 4-4.

PreProcessing		
Registration	PASS	Results
Android Manifest	PASS	Results
<hr/>		
Tools		
Test Tool 1	PASS	Results
Test Tool 2	WARNING	Results

Figure 4-4. Tool statuses.

A tool service status indicates the current processing status of, or the risk assessment generated by, a tool service. In general, the risk assessment generated by a tool service should conform to a standardized risk scoring system, such as the Common Vulnerability Scoring System (CVSS). AppVet also requires that these assessments be mapped to one of three risk categories: PASS, WARNING, or FAIL. Table 4-2 describes these risk assessment categories and AppVet tool service statuses.

Table 4-2. Tool status and risk assessment descriptions.

Type	Tool Status	Description
Processing Status	N/A	No status information is available for the tool.
	PENDING	App is waiting to be submitted to the tool service.
	SUBMITTED	App has been sent to the tool service.
	PROCESSING	Tool service is analyzing the app.
	ERROR	Tool service could not successfully analyze the app.
Risk Assessment	PASS	Tool service designates app as low-risk.
	WARNING	Tool service designates app as moderate-risk.
	FAIL	Tool service designates app as high-risk.

Note that a risk assessment is only displayed after the tool has successfully completed processing. Further note that risk assessments generated by a tool service can be later overridden if needed. This feature is used to mitigate against false positive analysis results.

4.2.4 App Status

In the apps list panel, each app is associated with a status or risk assessment as shown in Figure 4-5.



Figure 4-5. App status.

An app status indicates the current status or risk assessment of an app which, in turn, is based on the statuses and risk assessments of the tool services. Table 4-3 describes the AppVet app statuses and risk assessments.

Table 4-3. App status descriptions.

Type	App Status	Description
Processing Status	REGISTERING	App is being registered by AppVet.
	PENDING	App is waiting to be analyzed.
	PROCESSING	App is being analyzed by one or more tool services.
	ERROR	One or more tools could not analyze the app.
Risk Assessment	PASS	All tool services designated the app as low-risk.
	WARNING	At least one tool service designated the app as moderate-risk, but no tool designated the app as high-risk.
	FAIL	At least one tool service designated the app as high-risk.

Note that the decision to approve or reject an app is based on the overall risk assessment provided by AppVet is dependent upon the policies and security requirements of the organization that owns and operates the instance of AppVet. In addition, overall risk assessments generated by AppVet are based solely on the risk assessments provided by the tools used.

4.3 Managing Tools

AppVet 1.0 does not currently support the management of tool service adapters from the AppVet app management interface. However, this capability is expected in a future AppVet release.

5 DEVELOPER'S GUIDE

This section describes developing and building AppVet source code. Note that before editing, compiling, and running AppVet source code, required applications, components, and platforms must first be installed and configured as described in Section 2.

5.1 Eclipse

The AppVet source code release is distributed as an Eclipse IDE project and requires Eclipse v4.3 (Kepler) to be installed. For 64-bit operating systems, it is recommended to use 64-bit Eclipse. After installing Eclipse, install the Google Web Toolkit (GWT) plug-in by selecting **Help** → **Install New Software** from Eclipse. Next, select **Add...** and enter:

```
https://dl.google.com/eclipse/plugin/4.3
```

in the location field and select **Next>**. Then, select the following software packages to install:

- Google Plugin for Eclipse (required)
- GWT Designer for GPE (recommended)
- SDKs/Google Web Toolkit SDK 2.5.1 (Note that AppVet has not been tested using GWT 2.6.x)

After installing these software packages, restart Eclipse.

5.1.1 Integrating Tomcat

Development of AppVet requires that Tomcat be integrated with Eclipse. To integrate Tomcat with Eclipse, open Eclipse's **Servers** view by selecting **Windows** → **Show View** and select **Servers** from the **Server** folder. This will open a **Servers** pane in the Eclipse IDE. Next, right-click on the **Servers** pane and select **New** → **Server**. From the **New Server** dialog, select **Apache Tomcat v7.x** then select **Next**. From the **Add and Remove** dialog, add the **appvet** project to the **Configured** panel and select **Finish**. To save the modified configuration, select **File** → **Save** from the Eclipse menu.

You will see the newly created server in Eclipse's **Servers** pane. Next, double-click on the newly created server to open the server's configuration panel. Here, select **Use Tomcat installation** under **Server Locations** and set the **Deploy** path to `$CATALINA_HOME/webapps` directory. In addition, select **Serve modules without publishing** under **Server Options**.

5.1.2 Downloading AppVet

The AppVet source code distribution can be downloaded from GitHub at <https://github.com/AppVet/appvet>. Note that this source code release contains Eclipse `.project`, `.classpath`, and `.settings` files to facilitate importing the project into Eclipse. Future versions of this release, however, will remove these files and use Maven for building AppVet.

5.1.3 Importing AppVet

To import the `appvet` project in Eclipse, select **File → Import → General → Existing Projects into Workspace** from Eclipse and select the `/appvet` directory. If errors (denoted by a red box with an "X") are displayed on the `appvet` project, open the Eclipse Problems panel by selecting **Window → Show View → Problems**. Then, attempt to resolve errors by right-clicking on an error and selecting **Quick Fix**. As noted above, the AppVet source release currently includes Eclipse `.project`, `.classpath`, and `.settings` files and these might cause incompatibilities with your specific development environment. If so, check your Eclipse Java Build Path properties to ensure proper referencing of libraries, classes, and source files.

5.1.4 AppVet Compilation

To compile AppVet, right-click on the AppVet project and select **Google → GWT Compile**. Next, select the `appvet` project then **Compile**. Note that compiling using the GWT Compiler will also compile all non-GWT code including the AppVet servlet.

5.1.5 Installing AppVet

Before running AppVet from the source code distribution, AppVet files and database must first be installed and configured. The AppVet installer can be launched from Eclipse by right clicking on the `/src/gov/nist/appvet/installer/AppVetInstaller.java` file and selecting **Run As → Java Application**. Next, edit the AppVet properties file `$APPVET_FILES_HOME/conf/AppVetProperties.xml` and change the XPath value in `/Appvet/Logging/Level` from `INFO` to `DEBUG`. In addition, change the XPath value in `/Appvet/Logging/ToConsole` from `false` to `true`. These changes will set AppVet to display `DEBUG` log messages to the Eclipse console. For production use, reset these values back to their original values.

5.1.6 Launching AppVet

To launch AppVet from Eclipse's Servers panel, select the **Start the server** button on the Servers panel. As AppVet starts up, you will see output generated in the Eclipse console. To verify that AppVet has started, open a browser and enter the AppVet URL: `https://<host>:<port>/appvet`. At this point, the AppVet login page should be visible.

5.1.7 Exporting AppVet

To export AppVet to a production system, right-click on the `appvet` project in Eclipse and select **Export → WAR file** and select **Next**. Then, select `appvet` as the Web project and select the destination. To ensure optimal performance, select **Optimize for Apache Tomcat v7** under Target Runtime.

5.2 AppVet Architecture

To better understand the AppVet source code, it is necessary to understand the AppVet architecture. The AppVet architecture comprises three main components: AppVet Servlet, GWT Client/Server, and Tool Manager. These components are shown in Figure 5-1.

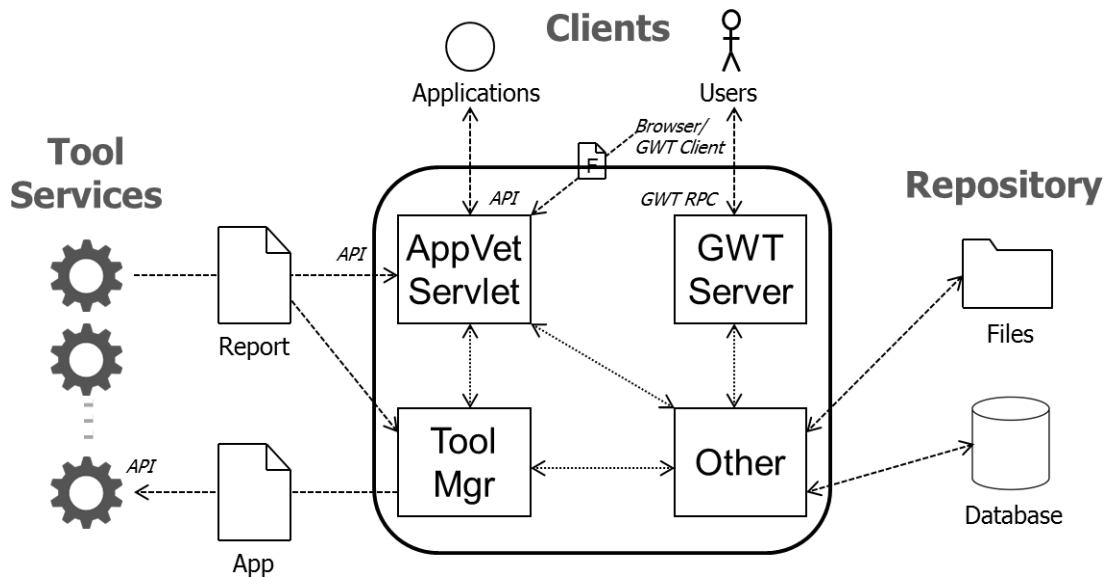


Figure 5-1. AppVet Architecture.

5.2.1 AppVet Servlet

The AppVet servlet implements the AppVet API for clients to interact with AppVet (see Appendix A). The AppVet servlet is used directly by applications including app stores and third-party applications, but is not directly accessed by users. Instead, users invoke the AppVet API via Google Web Toolkit (GWT) Client code within a web browser.

5.2.2 GWT Client/Server

The GWT Client provides the user interface to AppVet. Here, GWT provides the web browser widgets required for users to interact with AppVet. During development, GWT widgets are written in Java and compiled to AJAX for deployment. In Eclipse, use the GWT Designer to modify AppVet panels, layouts, and widgets. The GWT server supports requests from GWT client widgets including authentication and app info requests. The GWT client communicates via Remote Procedure Call (RPC) with the GWT server. During development, GWT servers are written in Java and compiled as Java classes for deployment. Note that a GWT server does not directly support file uploads from a GWT client and that file uploads from the AppVet GWT client must be sent directly to the AppVet servlet. Figure 5-1 shows a file *F* being uploaded to the AppVet servlet from a web browser containing the AppVet GWT client code.

5.2.3 Tool Manager

The tool manager is the AppVet component that manages the overall processing of an app by a set of tool services. The tool manager is responsible for extracting an app from the queue and

forwarding the app to the set of available tool services as defined by the set of AppVet tool service adapter files. The tool manager also processes reports from synchronous tools. Note that reports from asynchronous or push-type tools are processed by the AppVet servlet.

5.2.4 Other

Other AppVet components provide a wide variety of functionality from database transactions and file handling to input validation and logging.

5.3 AppVet Source Code Files

The AppVet source code release includes the files as shown in Figure 5-2.

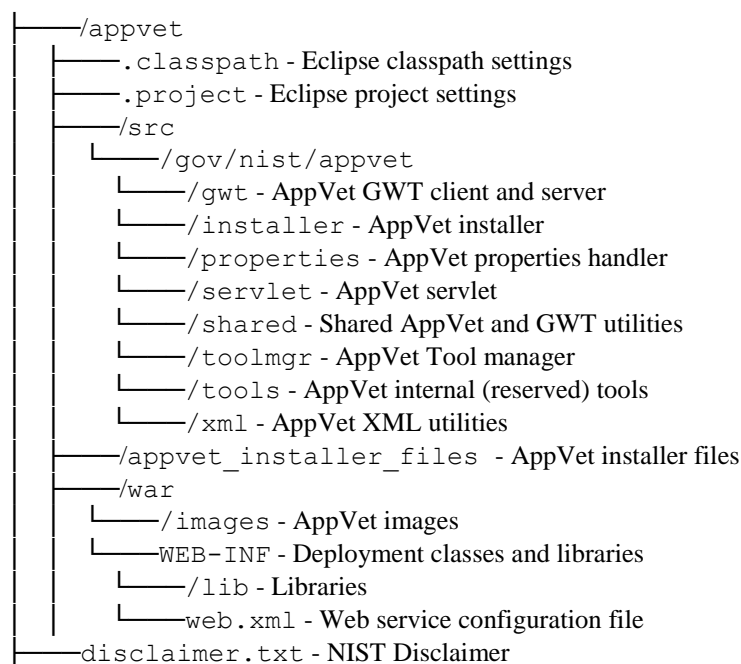


Figure 5-2. AppVet source code release.

The AppVet source code release comprises two main directories: `/src` and `/war`. The `/src` directory contains the AppVet source code while the `/war` directory contains the files required for deploying AppVet as a web application and service.

Note that it is strongly recommended to conduct AppVet development on a separate staging machine from the production host and to ensure correct operation of AppVet on the staging machine before being exported to the production host.

5.4 Database

The AppVet database, `appvet`, and tables are created during installation using the AppVet installer. Appvet tables include:

- `apps`: Defines app-related information including app name, owner, and app status.
- `status`: Defines tool statuses for each app.
- `users`: Defines user information.
- `sessions`: Defines user session information.

AppVet database schemas are defined in Appendix E.

5.5 Tool Services

The data that AppVet sends to a tool service is dependent on the tool service's RESTAPI. To be compatible with AppVet, a tool service's API must conform to the AppVet Tool Service Requirements as described in Appendix B. To facilitate integration of AppVet with a tool service, a tool service adapter is used by AppVet. This adapter should reflect the tool service's REST API, along with additional requirements specified in Appendix B. It is expected that tool service vendors, developers, or users that wish to wrap an existing tool as a service will provide the corresponding `ToolServiceAdapter.xml` file for their service. To define a `ToolServiceAdapter.xml` file, please consult the Tool Service Adapter XML schema defined in Appendix D.

In cases where a tool exists but no convenient method for accessing the tool is available, a Java servlet, PHP, or other type of service wrapper should be developed to support online access to the tool. Such a wrapper should implement a REST API that conforms to the requirements as described in Appendix B. An example tool service for verifying certificates of Android apps is available at <https://github.com/AppVet/android-cert-test>. This example includes source code that can be easily modified to support other tools and integrated with an AppVet system.

APPENDIX A: APPVET API

This section describes the AppVet services API specification. Items enclosed by '<' and '>' denote variable values.

A.1 AUTHENTICATE

The AUTHENTICATE service accepts a username and password for authenticating a client with AppVet and returns an AppVet session ID. This session ID may then be used by the client for further interaction with AppVet until the session expires.

A.1.1 HTTP Request

Entity	Name and/or Value	Description
Method	GET	HTTP Request method.
Parameter	command = AUTHENTICATE	AppVet command.
Parameter	username = <username>	AppVet username.
Parameter	password = <password>	AppVet password.

A.1.2 HTTP Response

Entity	Name and/or Value	Description
Status Code	<statuscode>	HTTP status code.
Payload	<sessionid>	AppVet session ID.

A.2 GET_STATUS

The GET_STATUS service retrieves the current status or risk assessment of an app. App status descriptions are described in Section 4.

A.2.1 HTTP Request

Entity	Name and/or Value	Description
Method	GET	HTTP Request method.
Parameter	command = GET_STATUS	AppVet command.
Parameter	sessionid = <sessionid>	AppVet session ID.
Parameter	appid = <appid>	AppVet app ID.

A.2.2 HTTP Response

Entity	Name and/or Value	Description
Status Code	<statuscode>	HTTP status code.
Payload	<appstatus>	AppVet app status or risk assessment.

A.3 GET_TOOL_REPORT

The GET_TOOL_REPORT service retrieves the tool report for the specified app.

A.3.1 HTTP Request

Entity	Name and/or Value	Description
Method	GET	HTTP Request method.
Parameter	command = GET_TOOL_REPORT	AppVet command.
Parameter	sessionid = <sessionid>	AppVet session ID.
Parameter	appid = <appid>	AppVet app ID.
Parameter	toolid = <toolid>	AppVet tool ID.

A.3.2 HTTP Response

Entity	Name and/or Value	Description
HTTP Status Code	<statuscode>	HTTP status code.
Payload	<reportfile>	Report file.

A.4 GET_APP_LOG

The GET_APP_LOG service retrieves the log for the specified app.

A.4.1 HTTP Request

Entity	Name and/or Value	Description
Method	GET	HTTP Request method.
Parameter	command = GET_APP_LOG	AppVet command.
Parameter	sessionid = <sessionid>	AppVet session ID.
Parameter	appid = <appid>	AppVet app ID.

A.4.2 HTTP Response

Entity	Name and/or Value	Description
Status Code	<statuscode>	HTTP status code.
Payload	<applogfile>	App log file.

A.5 GET_APPVET_LOG

The GET_APPVET_LOG service retrieves the AppVet system log.

A.5.1 HTTP Request

Entity	Name and/or Value	Description
Method	GET	HTTP Request method.
Parameter	command = GET_APPVET_LOG	AppVet command.
Parameter	sessionid = <sessionid>	AppVet session ID.

A.5.2 HTTP Response

Entity	Name and/or Value	Description
Status Code	<statuscode>	HTTP status code.
Payload	<appvetlogfile>	AppVet log file.

A.6 DOWNLOAD_REPORTS

The DOWNLOAD_REPORTS service retrieves a zipped file containing all reports and logs for the specified app.

A.6.1 HTTP Request

Entity	Name and/or Value	Description
Method	GET	HTTP Request method.
Parameter	command = DOWNLOAD_REPORTS	AppVet command.
Parameter	sessionid = <sessionid>	AppVet session ID.
Parameter	appid = <appid>	AppVet app ID.

A.6.2 HTTP Response

Entity	Name and/or Value	Description
Status Code	<statuscode>	HTTP status code.
Payload	<reportsfile>	AppVet reports file.

A.7 SUBMIT_APP

The SUBMIT_APP service submits an app to AppVet.

A.6.1 HTTP Request

Entity	Name and/or Value	Description
Method	POST	HTTP Request method.
Parameter	command = SUBMIT_APP	AppVet command.
Parameter	sessionid = <sessionid>	AppVet session ID.
Parameter	appfile = <appfile>	App file.

A.6.2 HTTP Response

Entity	Name and/or Value	Description
Status Code	<statuscode>	HTTP status code.
Response Header	appid = <appid>	The AppVet ID of the submitted app.

A.8 SUBMIT_REPORT

The SUBMIT_REPORT service submits (or overrides an existing) tool report for the specified app. This service is used by asynchronous and push services to submit reports as well as analysts for overriding existing tool reports.

A.6.1 HTTP Request

Entity	Name and/or Value	Description
Method	POST	HTTP Request method.
Parameter	command = SUBMIT_REPORT	AppVet command.
Parameter	sessionid = <sessionid>	AppVet session ID.
Parameter	toolid = <toolid>	AppVet tool ID.
Parameter	toolrisk = <toolrisk>	The risk assessment PASS, WARNING, FAIL. If the service could not process the app, then risk should have a value of ERROR.
Parameter	reportfile = <reportfile>	Report file.

A.6.2 HTTP Response

Entity	Name and/or Value	Description
Status Code	<statuscode>	An HTTP status code.

APPENDIX B: TOOL SERVICE API REQUIREMENTS

B.1 General Requirements

B.1.1 HTTP Request

AppVet requires tool services to implement a REST API. The API specification for HTTP Request messages must include the following:

Entity	Name and/or Value	Description
Method	POST	HTTP Request method.
Parameter	username = <username>	Tool service username (if required)
Parameter	password = <password>	Tool service password (if required)
Parameter	appfile = <appfile>	App file.

B.1.2 Additional Information

In addition, the following information must also be published by the tool service vendor:

Entity	Name/Value	Description
Service Name	<servicename>	Service name.
Service Description	<description>	Service description.
Service ID	<serviceid>	A lowercase alphabetic string used to distinguish this tool service from other tool services (e.g., "androwarn").
Report File Type	TXT/HTML/PDF/RTF	Report file type.
Service Type Protocol	SYNCHRONOUS/ ASYNCHRONOUS/ PUSH	Tool service type.

B.2 Synchronous Response

AppVet requires that synchronous services return the following in an HTTP Response message. Here, the risk assessment determined by the tool service should be defined as the named header `toolrisk` in the HTTP Response message. The risk assessment value `<toolrisk>` should be based on a standardized scoring system such as the Common Vulnerability Scoring System (CVSS) and mapped to one of three values: PASS, WARNING, or FAIL as described in Section 4.2.3. If an error occurs with the tool service, then ERROR should be returned as the `toolrisk` header value.

Entity	Name/Value	Description
Status Code	<code>	HTTP status code.
Response Header	toolrisk = <toolrisk>	The risk assessment.
Payload	<reportfile>	Report file.

B.3 Asynchronous Request

AppVet requires that asynchronous services support an `appid` parameter in an incoming HTTP Request message.

Entity	Type	Name/Value	Description
Parameter	String	<code>appid = <appid></code>	The AppVet ID of the submitted app. When returning reports and risk assessments back to AppVet, this ID should be included. See the <code>SUBMIT_REPORT</code> service in Appendix A.

AppVet requires that asynchronous services return an HTTP Response message with appropriate HTTP status code immediately after receiving an app. The HTTP status code should reflect the state of receipt of an app and related information. Because the service is asynchronous, a separate HTTP Request message must be used to return the service's reports and risk assessments (See `SUBMIT_REPORT` in Appendix A.8).

Entity	Type	Name/Value	Description
Status Code	String	<statuscode>	HTTP status code.

B.4 Asynchronous and Push Reports

AppVet requires that both asynchronous and push services return reports and risk assessments via an HTTP Request to AppVet. Here, the AppVet `SUBMIT_REPORT` service should be used (see Appendix A.8). Note that asynchronous and push services must have an AppVet username and password in order to authenticate with AppVet. Further note that asynchronous and push services do not require the use of an AppVet session ID for submitting a report and risk assessment for an app.

APPENDIX C: APPVET PROPERTIES SCHEMA

Listing C-1 specifies the AppVet properties schema. This schema is available as XML Schema (XSD) and Scalable Vector Graphic (SVG) files on the AppVet website <http://csrc.nist.gov/projects/appvet>.

Listing C-1. AppVetProperties XML Schema.

```
<?xml version="1.0" encoding="UTF-8"?>

<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://csrc.nist.gov/projects/appvet"
  xmlns:appvet="http://csrc.nist.gov/projects/appvet"
  elementFormDefault="qualified">

  <element name="AppVet">
    <annotation>
      <documentation>The AppVet element defines the properties of an AppVet system.
      </documentation>
    </annotation>
    <complexType>

      <sequence>
        <element name="Host" type="appvet:HostType" maxOccurs="1"
          minOccurs="1">
          <annotation>
            <documentation>The Host element defines properties of the AppVet
            host.</documentation>
          </annotation>
        </element>

        <element name="Logging" type="appvet:LoggingType"
          maxOccurs="1" minOccurs="1">
          <annotation>
            <documentation>The Logging element defines properties of the AppVet
            logger.</documentation>
          </annotation></element>

        <element name="Sessions" type="appvet:SessionType"
          maxOccurs="1" minOccurs="1">
          <annotation>
            <documentation>The Sessions defines properties of AppVet user
            sessions.</documentation>
          </annotation></element>
        <element name="Database" type="appvet:DatabaseType"
          maxOccurs="1" minOccurs="1">
          <annotation>
            <documentation>The Database element defines properties of the AppVet
            database.</documentation>
          </annotation></element>

        <element name="ToolServices" type="appvet:ToolServicesType"
          maxOccurs="1" minOccurs="1">
          <annotation>
            <documentation>The ToolServices element defines properties common
            to all AppVet tool service adapters. </documentation>
          </annotation>
        </element>

        <element name="Apps" type="appvet:AppsType" maxOccurs="1" minOccurs="1">
          <annotation>
```

```

        <documentation>The Apps element defines properties of AppVet
apps.</documentation>
    </annotation></element>
</sequence>

</complexType>
</element>

<complexType name="LoggingType">
    <sequence>
        <element name="Level" type="appvet:LevelType" maxOccurs="1" minOccurs="1">
            <annotation>
                <documentation>The Level element defines the level used for AppVet
logging. Level values include DEBUG, WARNING, INFO, ERROR. For example,
<Level>ERROR</Level>. The Level element should be set to DEBUG during AppVet development
and to INFO, WARNING, or ERROR for operational AppVet systems.</documentation>
            </annotation></element>
        <element name="ToConsole" type="boolean" maxOccurs="1" minOccurs="1">
            <annotation>
                <documentation>The ToConsole element defines whether to write logging
information to the console (true) or not (false). For example,
<ToConsole>true</ToConsole>. The ToConsole element should be set to true during AppVet
development and false for operational AppVet systems.</documentation>
            </annotation></element>
    </sequence>
</complexType>

<simpleType name="LevelType">
    <restriction base="string">
        <enumeration value="DEBUG" />
        <enumeration value="INFO" />
        <enumeration value="WARNING" />
        <enumeration value="ERROR" />
    </restriction>
</simpleType>

<complexType name="DatabaseType">
    <sequence>
        <element name="URL" type="anyURI" maxOccurs="1" minOccurs="1">
            <annotation>
                <documentation>The URL element defines the URL of the JDBC database
connector. For example, <URL>jdbc:mysql://localhost/appvet</URL>.</documentation>
            </annotation></element>
        <element name="UserName" type="string" maxOccurs="1" minOccurs="1">
            <annotation>
                <documentation>The UserName element defines the database username. For
example, <UserName>steve</UserName>.</documentation>
            </annotation></element>
        <element name="Password" type="string" maxOccurs="1" minOccurs="1">
            <annotation>
                <documentation>The Password element defines the database password. For
example, <Password>mypassword1234</Password>.</documentation>
            </annotation></element>
    </sequence>
</complexType>

<complexType name="AndroidType">
    <sequence>
        <element name="MinSDK" type="int" maxOccurs="1" minOccurs="1"></element>
        <element name="TargetSDK" type="int" maxOccurs="1" minOccurs="1"></element>
        <element name="MaxSDK" type="int" maxOccurs="1" minOccurs="0"></element>
    </sequence>
</complexType>

<complexType name="SessionType">
    <sequence>
        <element name="Timeout" type="int" maxOccurs="1"
minOccurs="1">
            <annotation>

```

```

        <documentation>The Timeout element defines the duration (in
milliseconds) of a user session. For example, <Timeout>1800000</Timeout>. Note that user
activity that is detected by the system resets Timeout.</documentation>
        </annotation>
    </element>
    <element name="GetUpdatesDelay" type="int" maxOccurs="1"
minOccurs="1">
        <annotation>
            <documentation>The GetUpdatesDelay element defines the interval (in
milliseconds) to retrieve session update information. For example,
<GetUpdatesDisplay>5000</GetUpdatesDisplay>.</documentation>
        </annotation>
    </element>
</sequence>
</complexType>

<complexType name="ServicesMgrType">
    <sequence>
        <element name="ConnectionTimeout" type="int" maxOccurs="1"
minOccurs="1"></element>
        <element name="SocketTimeout" type="int" maxOccurs="1" minOccurs="1"></element>
        <element name="Timeout" type="int" maxOccurs="1" minOccurs="1"></element>
    </sequence>
</complexType>

<complexType name="ToolServicesType">
    <sequence>
        <element name="PollingInterval" type="int" maxOccurs="1"
minOccurs="1">
            <annotation>
                <documentation>The PollingInterval element defines the interval (in
milliseconds) to poll for new or updated app information. For example,
<PollingInterval>2000</PollingInterval>.</documentation>
            </annotation>
        </element>
        <element name="StaggerInterval" type="int" maxOccurs="1"
minOccurs="1">
            <annotation>
                <documentation>The StaggerInterval element defines the delay (in
milliseconds) to invoke a tool on an app. For example,
<StaggerInterval>1000</StaggerInterval>.</documentation>
            </annotation>
        </element>
        <element name="ConnectionTimeout" type="int" maxOccurs="1"
minOccurs="1">
            <annotation>
                <documentation>The ConnectionTimeout element defines the maximum time
permitted (in milliseconds) to establish a connection to a remote tool service.
</documentation>
            </annotation>
        </element>
        <element name="SocketTimeout" type="int" maxOccurs="1"
minOccurs="1">
            <annotation>
                <documentation>The SocketTimeout element defines the maximum time
permitted (in milliseconds) to receive data from a remote tool service.</documentation>
            </annotation>
        </element>
        <element name="Timeout" type="int" maxOccurs="1" minOccurs="1">
            <annotation>
                <documentation>The Timeout element defines the maximum time permitted (in
milliseconds) for a synchronous service to complete.</documentation>
            </annotation></element>
    </sequence>
</complexType>

<complexType name="HostType">
    <sequence>
        <!-- Hostname must be a Fully Qualified Domain Name (FQDN)
that includes both a local

```

hostname and a domain name. This is required due to reliability issues in deriving the FQDN from the host IP. Because AppVet clients must connect to AppVet server(s) using the FQDN, it is necessary to require the admin to define this property. Note that if a FQDN is used for the host, all clients must also use the same FQDN when connecting to the host. Similarly, if an IP address in lieu of an FQDN for the host, all clients must connect to the host using that IP address.-->

```

<element name="Hostname" type="string" maxOccurs="1"
minOccurs="1">
  <annotation>
    <documentation>The Hostname element defines the fully qualified domain
name (FQDN) or IP address of the host. For example,
<Hostname>myhost.example.com</Hostname>. If DHCP is used, Hostame must have the value
Hostname="DHCP".</documentation>
  </annotation>
</element>
<element name="SSL" type="boolean" maxOccurs="1"
minOccurs="1">
  <annotation>
    <documentation>The SSL element defines whether SSL is used by the
Tomcat server (true) or not (false). For example, <SSL>true</SSL>. Note that the value
of SSL (true or false) must match the Tomcat server configuration as defined in
$CATALINA_HOME/conf/server.xml.</documentation>
  </annotation>
</element>
<element name="Port" type="string" maxOccurs="1"
minOccurs="1">
  <annotation>
    <documentation>The Port element defines the Tomcat port number for
accessing AppVet. For example, <Port>8080</Port>. Note that the value of Port must match
the Tomcat server configuration as defined in
$CATALINA_HOME/conf/server.xml.</documentation>
  </annotation>
</element>
<!-- Defines whether to keep received apps (true) or delete them
after processing (false). -->
</sequence>
</complexType>

<complexType name="AppsType">
  <sequence>
    <element name="KeepApps" type="boolean" default="false" maxOccurs="1"
minOccurs="1">
      <annotation>
        <documentation>The KeepApps element defines whether AppVet should keep
(true) or delete (false) received apps. Note that keeping apps on the system can lead to
large disk usage.</documentation>
      </annotation></element>
    </sequence>
  </complexType>
</schema>

```

APPENDIX D: TOOL SERVICE ADAPTER SCHEMA

Listing D-1 specifies the tool service adapter schema. This schema is available as XML Schema (XSD) and Scalable Vector Graphic (SVG) files on the AppVet website <http://csrc.nist.gov/projects/appvet>.

Listing D-1. Tool Service Adapter Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="
  http://csrc.nist.gov/projects/appvet"
  xmlns:appvet="http://csrc.nist.gov/projects/appvet"
  elementFormDefault="qualified"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <element name="ToolServiceAdapter">
    <annotation>
      <documentation>The ToolServiceAdapter defines properties for an AppVet tool
service.</documentation>
    </annotation>
    <complexType>
      <sequence>
        <element name="Description" type="appvet:DescriptionType"
          maxOccurs="1" minOccurs="1" >
          <annotation>
            <documentation>The Description element defines general information
about an AppVet tool service.</documentation>
          </annotation></element>
        <element name="Protocol" type="appvet:ProtocolType"
          maxOccurs="1" minOccurs="1" >
          <annotation>
            <documentation>The Protocol element defines protocol information for an
AppVet tool service.</documentation>
          </annotation></element>
      </sequence>
    </complexType>
  </element>

  <complexType name="DescriptionType">
    <sequence>
      <element name="Name" type="string" maxOccurs="1" minOccurs="1" >
        <annotation>
          <documentation>The Name element defines the name of an AppVet tool
service. For example, <Name>My Android Tester</Name>.</documentation>
        </annotation></element>
      <element name="Id" maxOccurs="1" minOccurs="1">
        <annotation>
          <documentation>The Id element defines the ID of the AppVet tool
service. The Id element must be alphabetic lowercase. For example,
<Id>myandroidtester</Id>.</documentation>
        </annotation>
        <simpleType>
          <restriction base="string">
            <pattern value="[a-z]+" />
          </restriction>
        </simpleType>
      </element>
      <element name="VendorName" type="string" maxOccurs="1"
        minOccurs="0" >
        <annotation>
          <documentation>The VendorName element defines the vendor name of the
AppVet tool service. For example, <Vendor>Example, Inc.</Vendor>.</documentation>
        </annotation></element>
    </sequence>
  </complexType>
</schema>
```

```

        <element name="VendorWebsite" type="anyURI" maxOccurs="1"
            minOccurs="0" >
            <annotation>
                <documentation>The VendorWebsite element defines the vendor's website. For
example, <VendorWebsite>https://www.example.com</VendorWebsite>.</documentation>
            </annotation></element>
        <element name="ReportFile" type="appvet:ReportFileType"
            maxOccurs="1" minOccurs="1">
            <annotation>
                <documentation>The ReportFile element defines the file type of the tool
service report. The ReportFormat element must be TXT, PDF, HTML, or RTF. For example,
<ReportFile>TXT</ReportFile>.</documentation>
            </annotation></element>
    </sequence>
</complexType>

<complexType name="ProtocolType">
    <annotation>
        <documentation>This type defines a remote AppVet-compatible tool
service
as synchronous or asynchronous.
    </documentation>
    </annotation>
    <sequence>
        <element name="Type" type="appvet:ProtocolNameType"
            maxOccurs="1" minOccurs="1">
            <annotation>
                <documentation>The Type element defines the service protocol for the
AppVet tool service. The Type element value must be SYNCHRONOUS, ASYNCHRONOUS, PUSH, or
INTERNAL.</documentation>
            </annotation>
        </element>
        <choice>
            <element name="Synchronous" type="appvet:SynchronousType" >
                <annotation>
                    <documentation>The Synchronous element defines the properties of an
AppVet synchronous tool service.</documentation>
                </annotation></element>
            <element name="Asynchronous" type="appvet:AsynchronousType" >
                <annotation>
                    <documentation>The Asynchronous element defines the properties of an
AppVet asynchronous tool service.</documentation>
                </annotation></element>
            <element name="Push" type="appvet:PushType" >
                <annotation>
                    <documentation>The Push element defines the properties of an AppVet
push tool service.</documentation>
                </annotation></element>
            <element name="Internal" type="appvet:InternalType" >
                <annotation>
                    <documentation>The Internal element defines the properties of an AppVet
internal tool.</documentation>
                </annotation></element>
        </choice>
    </sequence>
</complexType>

<simpleType name="ReportFileType">
    <restriction base="string">
        <enumeration value="TXT" />
        <enumeration value="HTML" />
        <enumeration value="PDF" />
        <enumeration value="RTF" />
    </restriction>
</simpleType>

<complexType name="SynchronousType">
    <annotation>
        <documentation></documentation>
    </annotation>
    <sequence>

```



```

    <element name="Request" type="appvet:HTTPRequestType"
      minOccurs="1" maxOccurs="1">
      <annotation>
        <documentation>The Request element defines an HTML Form to be sent to an
AppVet tool service via an HTTP Request message.</documentation>
      </annotation>
    </element>
    <element name="Response" type="appvet:HTTPResponseType"
      minOccurs="1" minOccurs="1" >
      <annotation>
        <documentation>The Response element defines properties for the response
from an AppVet synchronous tool service.</documentation>
      </annotation></element>
  </sequence>
</complexType>

<complexType name="AsynchronousType">
  <annotation>
    <documentation></documentation>
  </annotation>
  <sequence>
    <element name="Request" type="appvet:HTTPRequestType"
      minOccurs="1">
      <annotation>
        <documentation>The Request element defines an HTML Form to be sent to an
AppVet tool service via an HTTP Request message.</documentation>
      </annotation>
    </element>
    <element name="Response" type="appvet:HTTPResponseType"
      minOccurs="1" minOccurs="1" >
      <annotation>
        <documentation>The Reponse element defines the payload of an AppVet tool
service response. </documentation>
      </annotation></element>
  </sequence>
</complexType>

<complexType name="PushType">
  <sequence>
  </sequence>
</complexType>

<complexType name="InternalType">
  <sequence>
  </sequence>
</complexType>

<complexType name="HTTPRequestType">
  <annotation>
    <documentation />
  </annotation>
  <sequence>
    <element name="URL" type="anyURI" minOccurs="1" maxOccurs="1">
      <annotation>
        <documentation>The URL element defines the URL of the AppVet tool service.
      </documentation>
      </annotation>
    </element>
    <element name="Method" type="appvet:MethodType" maxOccurs="1"
      minOccurs="1" >
      <annotation>
        <documentation>The Action element defines the HTTP action to be invoked.
The Action element must have values GET or POST.</documentation>
      </annotation></element>
    <element name="Parameter" type="appvet:ParameterType"
      minOccurs="1" maxOccurs="unbounded">
      <annotation>
        <documentation>The FormParameter element defines the HTML form parameters
to be sent to the AppVet tool service. For asynchronous tool services, the form
parameters must include the parameter "appid" to allow the reports to be matched by
AppVet to their associated app.</documentation>

```

```

        </annotation>
      </element>
    </sequence>
  </complexType>

  <complexType name="AsynchronousResponseType">
    <sequence>
      <element name="ReportPayload" type="boolean" default="false"
        maxOccurs="1" minOccurs="1" >
        <annotation>
          <documentation>The ReportPayload element defines whether a report is
contained in the response from an AppVet asynchronous service. The value of ReportPayload
for asynchronous services must always be false.</documentation>
        </annotation></element>
    </sequence>
  </complexType>

  <complexType name="ParameterType">
    <sequence>
      <element name="Name" type="string">
        <annotation>
          <documentation>The Name element defines the name of the parameter.
</documentation>
        </annotation>
      </element>
      <element name="Value" type="string" default="DEFINED_AT_RUNTIME">
        <annotation>
          <documentation>The Value element defines the value of the parameter
element.</documentation>
        </annotation>
      </element>
    </sequence>
  </complexType>

  <simpleType name="MethodType">
    <restriction base="string">
      <enumeration value="POST" />
      <enumeration value="GET" />
    </restriction>
  </simpleType>

  <simpleType name="ProtocolNameType">
    <restriction base="string">
      <enumeration value="SYNCHRONOUS" />
      <enumeration value="ASYNCHRONOUS" />
      <enumeration value="PUSH" />
      <enumeration value="INTERNAL" />
    </restriction>
  </simpleType>

  <complexType name="HTTPResponseType">
    <sequence>
      <element name="AppVetRiskHeaderName" type="string"
        default="APPVET_TOOL_RESULT" maxOccurs="1" minOccurs="0">
        <annotation>
          <documentation>The AppVetRiskHeaderName element defines the
HTTP Response Header name for a synchronous tool
service's risk assessment category (ERROR, FAIL, WARNING,
or PASS). For synchronous services, the
AppVetResultHeaderName value should be
set to "toolrisk". For all service types, the AppVetRiskHeaderName
element should not be set.
          </documentation>
        </annotation>
      </element>
      <element name="ReportPayload" type="boolean" default="true"
        maxOccurs="1" minOccurs="1">
        <annotation>
          <documentation>
            The ReportPayload element defines whether the
            response message contains a report file (in

```

```

        string form). For synchronous tool services,
        this value must always be true.
    </documentation>
</annotation>
</element>
<element name="StatusCode" type="string" maxOccurs="unbounded" minOccurs="0">
    <annotation>
        <documentation>This is the HTTP status code returned by the tool service.
    </documentation>
    </annotation></element>
</sequence>
</complexType>
</schema>

```

APPENDIX E: APPVET DATABASE

This section defines the appvet database schema.

E.1 Table users

Table A. users schema.

Field	Type	Null	Key	Default	Extra
username	varchar(32)	NO	PRI		
password	varchar(102)	YES		NULL	
org	varchar(120)	YES		NULL	
email	varchar(120)	YES		NULL	
role	varchar(48)	YES		NULL	
lastlogon	timestamp	NO		CURRENT_TIMESTAMP	on update CURRENT_TIMESTAMP
fromhost	varchar(120)	YES		NULL	
lastName	varchar(32)	YES		NULL	
firstName	varchar(32)	YES		NULL	

Table B. Field descriptions.

Field	Description
username	The user's ID.
password	The user's password (PBKDF2 salted hash).
org	User's organization.
email	User's email.
role	User's role.
lastlogon	Time of last logon.
fromhost	Host of last logon.
lastName	User's last name.
firstName	User's first name.

E.2 Table sessions

Table A. sessions schema.

Field	Type	Null	Key	Default	Extra
sessionid	varchar(32)	NO	PRI	NULL	
username	varchar(120)	YES		NULL	
clientaddress	varchar(120)	YES		NULL	
expiretime	bigint(20)	YES		NULL	

Table B. Field descriptions.

Field	Description
sessionid	Session ID.
username	User ID.
clientaddress	Client host.
expiretime	Time of session expiration.

E.3 Table apps**Table A. apps schema.**

Field	Type	Null	Key	Default	Extra
appid	varchar(32)	NO	PRI		
appname	varchar(120)	YES		NULL	
packagename	varchar(120)	YES		NULL	
versioncode	varchar(120)	YES		NULL	
versionname	varchar(120)	YES		NULL	
filename	varchar(120)	YES		NULL	
submittime	timestamp	YES		NULL	
status	varchar(120)	YES		NULL	
statustime	timestamp	YES		NULL	
username	varchar(120)	YES		NULL	
clienthost	varchar(120)	YES		NULL	

Table B. Field descriptions.

Field	Description
appid	AppVet-generated unique identifier.
appname	The name of the app.
packagename	The Android package name of the app.
versioncode	The version code of the app.
versionname	The version name of the app.
filename	The file name of the app.
submittime	The time the app was submitted.
status	The current status of the app.
statustime	The time of the last status update.
username	The client who submitted the app.
clienthost	The host who submitted the app.

E.4 Table status

The fields in the `status` table are dependent upon the set tools available to AppVet.

Table A. status schema.

Field	Type	Null	Key	Default	Extra
appid	varchar(32)	NO	PRI		
registration	varchar(120)	YES		NULL	
android	varchar(120)	YES		NULL	

Table B. Field descriptions.

Field	Description
appid	AppVet-generated unique identifier.
registration	Registration status.
android	Android pre-processing status.

APPENDIX F: CHANGE HISTORY

7/14/14

- Recommended 64-bit Windows as the host OS due to problems with Linux installation.
- Removed Linux installation instructions.

5/28/14

- AppVet now returns an app ID in response to a `SUBMIT_APP` command. The app ID is defined by the `appid` header in the corresponding HTTP Response message. This change only affects clients that programmatically interact with AppVet including app stores or continuous integration environments.